PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-029787
(43)Date of publication of application : 28.01.2000

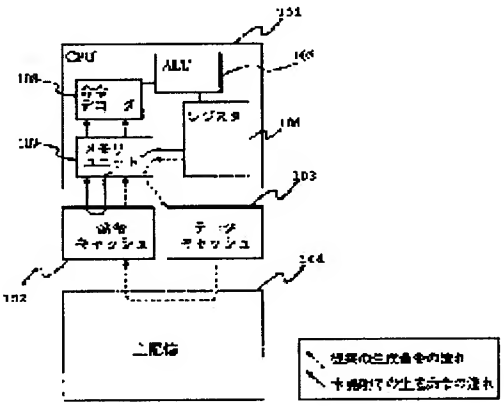
(51)Int.Cl. G06F 12/08
G06F 9/45

(21)Application number : 10-200266
(22)Date of filing : 15.07.1998

(71)Applicant : HITACHI LTD
(72)Inventor : NISHIYAMA HIROYASU

(54) PROCESSOR EQUIPPED WITH WRITING-IN MECHANISM FOR INSTRUCTION CACHES

(57)Abstract:
PROBLEM TO BE SOLVED: To suppress low the decrease in performance due to main storage reference at dynamic instructing generating by providing in an instruction set an instruction, capable of specifying by each memory area as an area which is written through a data cache or an area which is written through an instruction cache.
SOLUTION: In the instruction set, the instruction is provided which can specify each memory area as an area which is written via the data case or an area which is written through the instruction cache. This processor writes an instruction generated dynamically by ALU 105 directly to the instruction cache 102 through a memory unit 107, after it is stored in a register 106. The generated instruction, when executed is read out of the instruction cache 102 by a memory unit 107 and passed to an instruction decoder 108. Thus, the instruction written to the instruction cache 102 can be executed directly not through a main storage 104, so that the execution of the dynamically generated instruction can be speeded up.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－29787
（P2000－29787A）
(43)公開日　平成12年１月28日(2000.1.28)

(51)Int.Cl.[7]	識別記号	ＦＩ		テーマコード（参考）
Ｇ０６Ｆ　12/08		Ｇ０６Ｆ　12/08	Ｇ	５Ｂ００５
			Ｗ	５Ｂ０８１
	３１０		３１０Ｂ	
9/45		9/44	３２０Ｃ	

審査請求　未請求　請求項の数７　ＯＬ　（全 11 頁）

(21)出願番号　特願平10－200266

(22)出願日　平成10年７月15日(1998.7.15)

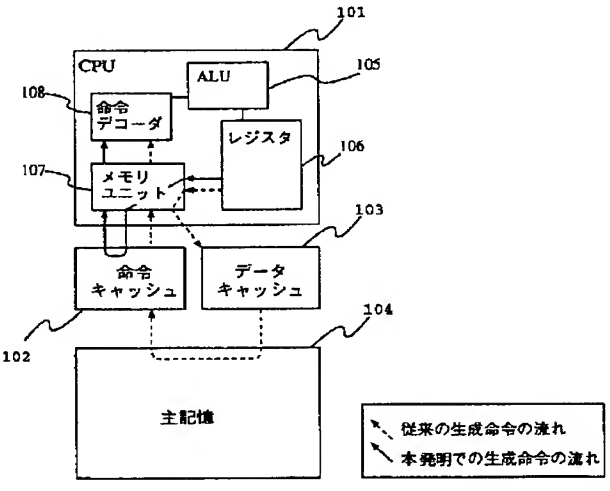
(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　西山　博泰
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内
(74)代理人　100087170
弁理士　富田　和子

Ｆターム(参考)　5B005 JJ11 KK13 MM02 MM03 MM05 UU41
5B081 DD00

(54)【発明の名称】　命令キャッシュへの書き込み機構を備えたプロセッサ

(57)【要約】
【課題】　命令キャッシュとデータキャッシュを分離した非統合型キャッシュを採用したプロセッサにおいて、仮想マシンインタプリタのＪＩＴコンパイラ等によって実行時に動的に生成された命令を、一旦主記憶や下位レベルのキャッシュメモリといった低速な記憶装置を介して命令キャッシュに読み出す必要をなくし、動的に生成した命令を命令キャッシュに高速に反映できるようにする。
【解決手段】　マイクロプロセッサに対して、（ａ）所定のメモリ領域毎に、データキャッシュと命令キャッシュのいずれを介して書き込みを行う領域かを指定可能とする、（ｂ）データキャッシュと命令キャッシュのいずれを介して書き込みを行うかを指定するフラグを設ける、又は（ｃ）データキャッシュを介してメモリへの書き込みを行う命令と命令キャッシュを介してメモリへの書き込みを行う命令を提供する。

図1



1

【特許請求の範囲】

【請求項１】　命令セットの中に、所定のメモリ領域毎に、当該領域が、データキャッシュを介して書き込みを行う領域か、命令キャッシュを介して書き込みを行う領域かを指定することができる命令を有することを特徴とするプロセッサ。

【請求項２】　前記命令によって命令キャッシュを介して書き込みを行う領域として指定されたメモリ領域に対して、書き込みが指示された場合は、命令キャッシュを介して書き込みを行うことを特徴とする請求項１に記載のプロセッサ。

【請求項３】　データキャッシュを介して書き込みを行うモードであるか、命令キャッシュを介して書き込みを行うモードであるかを表すフラグを備え、
当該フラグが命令キャッシュを介して書き込みを行うモードであることを示している場合は、命令キャッシュを介して書き込みを行うことを特徴とするプロセッサ。

【請求項４】　命令セットの中に、前記フラグを操作する命令を有することを特徴とする請求項３に記載のプロセッサ。

【請求項５】　命令セットの中に、命令キャッシュを介してメモリへの書き込みを行うことを指示する命令を有することを特徴とするプロセッサ。

【請求項６】　命令キャッシュを介した書き込みが指示された場合、命令キャッシュとデータキャッシュの両方に書き込みを行うことを特徴とする請求項１乃至請求項５のいずれか一項に記載のプロセッサ。

【請求項７】　請求項１乃至請求項６のいずれか一項に記載のプロセッサを使った動的コンパイルの方法であって、
動的に生成した命令をメモリへ書き込む場合、命令キャッシュを介して書き込むようにすることを特徴とする動的コンパイルの方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、プロセッサに関し、特に、実行時に動的に機械語命令プログラムを生成して実行を行う仮想マシンインタプリタ等を実行するのに適したマイクロプロセッサに関する。

【０００２】

【従来の技術】従来のマイクロプロセッサでは、主記憶（メインメモリ）とＣＰＵとの間に、高速な中間記憶装置として、キャッシュを設けることが多い。このキャッシュは、その構成上、データキャッシュと命令キャッシュを分離した非統合型キャッシュと、データキャッシュと命令キャッシュを統合した統合型キャッシュとに分類することができる。前者の非統合型キャッシュは、後者の統合型キャッシュと比較して、データキャッシュと命令キャッシュのそれぞれに、独立に高いデータ供給バンド幅を与えることができるので、一般に高い性能を得ることができる。このため、ハイエンドのマイクロプロセッサでは、非統合型のキャッシュが利用されることが多い。

2

【０００３】一方、このようなマイクロプロセッサで実行されるプログラムを記述するプログラミング言語として、Ｊａｖａ（Ｊａｖａは、米国 Sun Microsystems, Inc. の商標）やＳｍａｌｌｔａｌｋといったものが知られている。これらのプログラミング言語で記述されたプログラムは、バイトコードと呼ばれる仮想的な計算機の命令に一旦変換され、バイトコードは、通常、仮想マシンインタプリタと呼ばれるソフトウェアによって実行される。

【０００４】しかし、このような仮想マシンインタプリタによる実行は、仮想マシン命令の解釈実行などのオーバヘッドが大きい。従って、ＪａｖａやＳｍａｌｌｔａｌｋで記述されたプログラムは、ＣやＦＯＲＴＲＡＮといった従来のコンパイラ型言語で記述された場合に比べて、一般に、その実行速度は低速になる。これに対して、例えば、L.P. Deutsch と A.M. Chiffmanによる”Efficient Implementation of the Smalltalk-80 System”（In Proceedings of the 11th Annual ACM Symposium on Principles of Programming Languages pp. 297-302, 1984）に示されているように、実行対象の仮想マシン命令列に対して、それと等価な処理を行う機械語命令を実行時に生成し、生成した機械語命令を直接実行することでインタプリタ実行の高速化を行う手法が用いられる。このような手法は、機械語命令を実行時に動的に生成することから動的コンパイル、または実行を行おうとした時点でコンパイルを行うことからＪＩＴ（ジャストインタイム：Just-In-Time）コンパイルと呼ばれ、Ｊａｖａ仮想マシンの高速化などに利用されている。

【０００５】

【発明が解決しようとする課題】このような実行時に命令を動的に生成するＪＩＴコンパイル技術は、仮想マシンインタプリタなどの高速化には有効であるが、非統合型のキャッシュ機構を採用したマイクロプロセッサで利用することを考えた場合、実行時に生成された命令を、実行時に主記憶へ書き込むことが必要になることから、次のような問題が生じることになる。

【０００６】動的に生成された命令の主記憶への書き込みは、プロセッサから見た場合、通常のデータの書き込みと変わらないので、非統合型キャッシュを採用したプロセッサでは、データキャッシュを介して行われることになる。このため、生成された命令をデータキャッシュを介してメモリへ書き込んだ直後に、書き込んだ当該命令を実行しようとしても、命令キャッシュとデータキャッシュの整合性が保たれていないので、そのまま実行することができない。すなわち、命令キャッシュの更新が行われていないために、そのままでは予期しない命令を実行してしまう可能性がある。そこで、ほとんどのマイ

クロプロセッサでは、指定したデータキャッシュ上のデータを主記憶に書き戻したり、命令キャッシュ上のキャッシュブロックを無効化するための命令を備えている。そして、主記憶へ書き込んだ命令を実行する前に、このような命令を使って、書き込んだアドレスに対応する命令キャッシュ上のデータを破棄（無効化）するとともに、ライトバック（write-back）型のキャッシュであれば、変更が主記憶に反映されるように、書き込んだアドレスに対応するデータキャッシュ上のデータを主記憶に書き戻す。

【０００７】つまり、動的な命令の生成・実行処理は、以下のような流れで行われる。

【０００８】１．書き込み対象アドレス（以下、アドレスＡという）に対応するデータキャッシュのキャッシュブロックへの、動的に生成された機械語命令の書き込み
２．アドレスＡに対応する命令キャッシュのキャッシュブロックの破棄
３．アドレスＡに対応するデータキャッシュのキャッシュブロックの主記憶への書き戻し（ライトバック型のキャッシュの場合）
４．主記憶のアドレスＡから命令キャッシュの対応するキャッシュブロックへの読み出し
５．命令キャッシュからの命令の読み出し及び実行

以上述べたように、非統合型キャッシュを採用したマイクロプロセッサにおいて、実行時に動的に命令を生成し、当該生成された命令を実行しようとすると、動的に生成された命令は、データキャッシュに書き込まれた後、主記憶へ書き出され、その後、命令キャッシュへ読み出す必要があるため、主記憶への書き込みや読み出しを伴うことになり、このことは、実行速度の低下要因となりうる。

【０００９】特に、プロセッサ内での命令レベル並列度や動作周波数が向上し、主記憶参照に要する時間で、多数の命令を処理可能なハイエンドマイクロプロセッサでは、主記憶参照に要するオーバヘッドにより動的な命令生成処理を行うプログラムの動作性能が低下してしまうといった問題が深刻化するおそれがある。

【００１０】なお、ＣやＦＯＲＴＲＡＮといった従来の静的なコンパイラ型言語では、機械語命令への変換がプログラム実行前にあらかじめ行われ、実行時に動的に命令を生成することがないため、分離型のキャッシュ構成であっても、大きな問題とはならなかった。本発明の目的は、仮想マシンインタプリタにおけるＪＩＴコンパイルのような、実行時に動的に機械語命令の生成を行う処理において、動的な命令生成時の主記憶参照によって生じる性能低下を低く押えることができるプロセッサを提供することにある。

【００１１】

【課題を解決するための手段】本発明に係るプロセッサは、命令セットの中に、所定のメモリ領域毎に、当該領域が、データキャッシュを介して書き込みを行う領域か、命令キャッシュを介して書き込みを行う領域かを指定することができる命令を有することを特徴とする。この場合、前記命令によって命令キャッシュを介して書き込みを行う領域として指定されたメモリ領域に対して、書き込みが指示された場合は、命令キャッシュを介して書き込みを行う。

【００１２】また、本発明に係る第２のプロセッサは、データキャッシュを介して書き込みを行うモードか、命令キャッシュを介して書き込みを行うモードであるかを表すフラグを備え、当該フラグが命令キャッシュを介して書き込みを行うモードであることを示している場合は、命令キャッシュを介して書き込みを行うことを特徴とする。この場合、命令セットの中に、前記フラグを操作する命令を有することが好ましい。

【００１３】また、本発明に係る第３のプロセッサは、命令セットの中に、命令キャッシュを介してメモリへの書き込みを行うことを指示する命令を有することを特徴とする。

【００１４】なお、上記第１から第３のプロセッサにおいて、命令キャッシュを介した書き込みが指示された場合、命令キャッシュとデータキャッシュの両方に書き込みを行うようにしてもよい。このようにすれば、データキャッシュに命令キャッシュの更新データのコピーが存在することになるので、命令キャッシュからメモリへの直接的な書き戻し経路を設けることなく、命令キャッシュにおいてキャッシュ溢れが生じた場合の更新データの主記憶への書き戻しが可能になる。

【００１５】更に、本発明に係る動的コンパイルの方法は、前述の第１から第３のプロセッサを使った動的コンパイルの方法であって、動的に生成した命令をメモリへ書き込む場合、命令キャッシュを介して書き込むようにすることを特徴とする。

【００１６】本発明によるプロセッサでは、命令キャッシュを介してメモリへの書き込みを行うことができるので、動的コンパイルによって、動的に生成した命令をメモリへ書き込む場合、命令キャッシュを介して書き込むことができ、主記憶参照の必要性が減り、仮想マシンインタプリタにおけるＪＩＴコンパイルのような、実行時に動的に機械語命令の生成を行う処理において、動的な命令生成時の主記憶参照によって生じる性能低下を低く押えることができる

【００１７】

【発明の実施の形態】以下、本発明の実施の形態について、図面を参照しつつ、詳細に説明する。

【００１８】図９は、本発明を実施する計算機システムの例を示す図である。仮想マシンインタプリタにおけるＪＩＴコンパイラなど実行時に命令生成処理を行うソフトウェアは、ディスク装置２０４から主記憶２０３に読み出され、プロセッサ２０１によって実行される。この

ソフトウェアにより生成される機械語命令は、高速な中間記憶装置であるキャッシュ２０２を介して主記憶２０３とプロセッサ２０１の間で読み書きされる。なお、ここでは、簡単のためメモリ階層をキャッシュと主記憶の２階層としているが、本発明は、複数のキャッシュ階層を持つマイクロプロセッサに対して適用することも可能である。

【００１９】図１は、本発明による非統合型キャッシュを採用したマイクロプロセッサの動作概要を説明する図である。本発明と対比するため、まず、非統合型キャッシュを採用した従来のマイクロプロセッサの動作について同図を使って説明する。非統合型キャッシュを採用した従来のマイクロプロセッサでは、ＣＰＵ１０１が実行する命令は、主記憶１０４から命令キャッシュ１０２へ読み出され、ＣＰＵ１０１へと受け渡される。このようにしてフェッチされた命令は、メモリユニット１０７を介して、命令デコーダ１０８に渡される。更に、当該命令は、命令デコーダ１０８でデコードされて、当該デコード結果に基づいて、ＡＬＵ１０５やレジスタ１０６が制御されて、命令の実行が行われる。頻繁に実行される命令は、高速な中間記憶装置である命令キャッシュ１０２に保存されることにより、低速な主記憶１０４を参照する必要がなくなるため、高速な実行を行うことが可能になる。

【００２０】図１において、破線は、従来のマイクロプロセッサにおいて、動的命令生成処理での動的に生成された命令の流れを表す。また、実線は本発明における動的に生成された命令の流れを表している。なお、ここでの説明では、簡単のため、まず、キャッシュブロックの競合や容量不足による溢れが生じない場合について説明し、キャッシュブロックの溢れが発生する場合については後述する。

【００２１】同図の破線で示すように、従来の動的命令生成処理においては、ＡＬＵ１０５によって実行時に動的に生成された命令は、レジスタ１０６に格納された後、メモリユニット１０７を介して、データキャッシュ１０３へ書き込まれる。データキャッシュ１０３がライトバック方式を採用している場合、この状態では、実行すべき命令がデータキャッシュ１０３上にのみ存在するので、当該命令が書き込まれたキャッシュブロックを主記憶１０４へ書き戻す。また、命令キャッシュ１０２上に当該書き込みを行ったアドレスに対応するキャッシュブロックが存在する可能性があるので、存在する場合には命令キャッシュ１０２上の当該ブロックを無効化する。これによって、動的に生成された命令の書き込みを行ったアドレスに対応するキャッシュブロックは、命令キャッシュ１０２上に存在しなくなるので、生成された命令を実行しようとした時点で、主記憶１０４から命令キャッシュ１０２上に命令が読み出され、メモリユニット１０７を介して命令デコーダ１０８へと受け渡され、デコード結果に従って、ＣＰＵ１０１での命令実行が行われる。

【００２２】これに対して、本発明における動的命令生成処理での生成命令の流れでは、ＡＬＵ１０５が動的に生成した命令は、レジスタ１０６に格納された後、メモリユニット１０７を介して、命令キャッシュ１０２に直接書き込まれる。そして、生成された命令を実行する場合には、命令キャッシュ１０２上に生成された命令が存在するため、メモリユニット１０７は、命令キャッシュ１０２から実行すべき命令を読み出し、読み出した命令を命令デコーダ１０８へ渡す。

【００２３】以上説明したように、従来のマイクロプロセッサでは、キャッシュ容量の余裕の有無に関わらず、データキャッシュ１０３に書き込んだ命令を一度高速なキャッシュメモリから低速な主記憶１０４へ書き込んで、再度命令キャッシュ１０２を介してＣＰＵ１０１へ読み出さねばならなかった。これに対して、本発明によるマイクロプロセッサでは、主記憶１０４を介することなく直接命令キャッシュ１０２上に書き込んだ命令を実行することができるので、動的に生成された命令の実行を高速化することができる。

【００２４】以上の説明では、キャッシュブロックの競合や容量不足によるキャッシュブロックの追い出しを考慮していなかった。実際には、キャッシュ容量は限られており、キャッシュブロックの競合は生じるため、追い出し時の処理を考慮する必要がある。そこで、以下ではキャッシュ追い出し時の処理を考慮した命令書き込み処理について説明する。

【００２５】キャッシュへの書き込み処理において、競合等によりキャッシュブロックを追い出す場合の処理は、キャッシュがライトスルー（write-through）方式を採用しているか、ライトバック方式を採用しているかによってその処理が異なる。

【００２６】ライトスルー方式の場合には、キャッシュへの書き込みと同時に主記憶への書き込みも行われているため、追い出し対象のキャッシュブロックを単に無効化するだけで良い。すなわち、本発明による命令キャッシュをライトスルー方式のキャッシュとして構成した場合は、命令キャッシュと主記憶の両方に、書き込み対象の命令が書き込まれるので、主記憶上に最新のデータが保存されていることになり、キャッシュブロックの追い出し時にはキャッシュブロックの内容を破棄すれば良い。

【００２７】一方、ライトバック方式の場合には、最新のデータはキャッシュ上のみに存在するので、更新データを含むキャッシュブロックの追い出し時には主記憶へのデータの書き戻し処理を行う必要がある。従来のプロセッサでは、ＣＰＵからキャッシュへのデータの書き込みはデータキャッシュに対してのみ行われていたため、キャッシュから主記憶へのデータの書き戻し処理はデー

タキャッシュだけでしか行われていなかった。本発明によるプロセッサでは、命令キャッシュに対する書き込みも行うので、ライトバック方式を採用した場合、命令キャッシュからの主記憶へのデータの書き戻し処理を考慮する必要がある。このような命令キャッシュからの主記憶へのデータの書き戻し処理には、命令キャッシュから直接主記憶への書き込みを可能とする方式と、従来通り主記憶への書き込みはデータキャッシュのみから行うようにする方式とが考えられる。以下では、各方式についてその実現例を示す。

【００２８】図２は、命令キャッシュから主記憶へキャッシュブロックを書き込み可能なようにアーキテクチャを拡張した場合の、動的に生成された命令の流れを表している。図２において、ＡＬＵ１０５によって動的に生成され、レジスタ１０６に格納された命令は、メモリユニット１０７を介して命令キャッシュ１０２に書き込まれる。ここで、書き込み対象となるキャッシュブロック３０１に書き込みアドレスとは異なるアドレスに対応する命令が既に保持されている場合、そのキャッシュブロックを利用可能とする必要があるが、当該キャッシュブロック３０１の更新状態によってその処理が異なる。キャッシュブロック３０１が書き込みが行われていない状態（クリーン：clean）であれば、主記憶１０４上の内容と命令キャッシュ１０２上の内容は一致しているので、主記憶１０４への書き戻しは行わず、命令キャッシュ１０２上のキャッシュブロック３０１のデータを破棄する。一方、キャッシュブロック３０１が書き込みが行われた状態（ダーティ：dirty）であれば、命令キャッシュ１０２上のみに最新の命令が存在するので、まず、キャッシュブロック３０１を対応する主記憶上の領域３０２へ書き戻す。

【００２９】命令キャッシュ１０２上の元のデータの破棄または書き戻しが終了すると、動的に生成された命令の書き込み対象アドレスに対応した主記憶１０４上のデータブロック３０３をキャッシュブロック３０１へ読み込んで、メモリユニット１０７から受け渡された書き込み対象命令の書き込みを行う。これらの動作は、通常のライトバック型データキャッシュの動作と同様である。

【００３０】図３は、図２に示した命令キャッシュから主記憶へのキャッシュブロックの書き戻し機構を追加した場合の主記憶への書き込み処理の流れを示す図である。同図に示すように、処理を開始すると（Ｓ４０１）、まず、書き込みが命令キャッシュを介した書き込みか否かを判別する（Ｓ４０２）。あるデータ書き込み命令がデータキャッシュを介した書き込みか、命令キャッシュを介した書き込みかを指定・判断する方法については、後述する。

【００３１】判別の結果、当該書き込みがデータキャッシュを介したものであれば（Ｓ４０２：ＮＯ）、従来と同じデータキャッシュを介した書き込み処理を行い（Ｓ４０３）、処理を終了する（Ｓ４０８）。一方、命令キャッシュを介した書き込みであれば（Ｓ４０２：ＹＥＳ）、書き込み対象アドレスに対応するキャッシュブロックが命令キャッシュ上に存在するか否かを調べる（Ｓ４０４）。その結果、書き込み対象アドレスに対応するキャッシュブロックが命令キャッシュ上に存在しない場合は（Ｓ４０４：ＮＯ）、続けて、データキャッシュに書き込み対象アドレスに対応するキャッシュブロックが存在するか否かを調べ、そのようなキャッシュブロックが存在し、データキャッシュの当該キャッシュブロックがダーティ状態であれば当該ブロックを主記憶に書き出し（Ｓ４０５）、書き込み対象のアドレスに対応したブロックを命令キャッシュに読み出す（Ｓ４０６）。これにより、命令キャッシュ上に書き込み対象アドレスのキャッシュブロックが確保されるので、命令キャッシュの当該ブロックに動的に生成された命令を書き込み（Ｓ４０７）、処理を終える（Ｓ４０８）。一方、書き込み対象アドレスに対応するキャッシュブロックが命令キャッシュ上に存在する場合は（Ｓ４０４：ＹＥＳ）、命令キャッシュ上に書き込み対象アドレスのキャッシュブロックが既に確保されているので、そのまま、命令キャッシュに命令を書き込み（Ｓ４０７）、処理を終える（Ｓ４０８）。

【００３２】なお、命令書き込み対象アドレスに対応するキャッシュブロックがデータキャッシュ上に存在する場合は、命令キャッシュとの整合性を保つため、該当するデータキャッシュのデータブロックを破棄（無効化）するか、又は、命令キャッシュとともに、データキャッシュにも命令を書き込むようにすればよい。

【００３３】図４は、命令キャッシュ１０２から主記憶１０４へ直接書き込みを行う機構を有しない場合の、動的に生成された命令の流れを表している。この場合、命令キャッシュから主記憶への直接的な書き戻し経路を設ける必要がないので、図２に示したものに比べて、従来のマイクロプロセッサに対するハードウェア的な変更量が少なくてすむ。

【００３４】図４に示すように、ＡＬＵ１０５によって動的に生成され、レジスタ１０６に格納された命令は、メモリユニット１０７を介して命令キャッシュ１０２と、データキャッシュ１０３に書き込まれる。すなわち、データキャッシュ１０３上にも当該命令のコピーを持つこととする。

【００３５】このようにデータキャッシュ１０３上にも当該命令（に対応するキャッシュブロック）のコピーを持つのは、本実施形態では、命令キャッシュ１０２から主記憶１０４へのキャッシュブロックの書き戻しパスが存在しないため、命令キャッシュ１０２に対して動的に生成された命令の書き込みを行う際に、キャッシュ溢れが生じ、主記憶への書き戻しが必要になった場合、当該書き戻しをできるようにするためである。

【００３６】命令キャッシュ１０２への書き込みを行う際、キャッシュ溢れが生じ、主記憶への書き戻しが必要になった場合は、書き戻し対象となった命令キャッシュ上のブロック５０１を破棄するとともに、対応するデータキャッシュ１０３上のブロック５０２の更新状況を調べて、ダーティであれば、対応する主記憶１０４上の領域５０４に書き戻す。次に、書き込み対象アドレスに対応するブロック５０３を主記憶１０４から読み出し、命令キャッシュ１０２とデータキャッシュ１０３の両方に読み込む。そして、書き込み対象の命令を命令キャッシュ１０２上のブロック５０１と、データキャッシュ１０３上のブロック５０２へ書き込む。

【００３７】なお、命令キャッシュ１０２上のデータに対応して、データキャッシュ１０３上に確保されているキャッシュブロックは、データキャッシュへのデータの書き込みによる競合によって主記憶へ書き戻されることがあるが、その場合には、命令キャッシュ１０２上のキャッシュブロックの内容と主記憶１０４上のデータは一致しているので、特別な操作を行う必要はない。

【００３８】図５は、図４に示した命令キャッシュから主記憶へのキャッシュブロックの書き戻し機構を追加しない場合の主記憶への書き込み処理の流れを示す図である。同図に示すように、主記憶への書き込みを行う場合（Ｓ６０１）、まず、命令キャッシュを介した書き込みか否かのチェックを行う（Ｓ６０２）。書き込みが命令キャッシュを介した書き込みか否かの指定・判定方法については後述する。

【００３９】チェックの結果、書き込みが、命令キャッシュを介した書き込みでなければ（Ｓ６０２：ＮＯ）、通常のデータキャッシュを介した書き込み処理を行い（Ｓ６０３）、書き込み処理を終了する（Ｓ６１０）。

【００４０】一方、命令キャッシュを介した書き込みであれば（Ｓ６０２：ＹＥＳ）、まず、データキャッシュに書き込み対象アドレスに対応するブロックが存在するか否かを調べる（Ｓ６０４）。その結果、対応するブロックがデータキャッシュに存在しなければ（Ｓ６０４：ＮＯ）、データキャッシュに当該ブロックを読み出す（Ｓ６０５）。更に、命令キャッシュに書き込み対象アドレスに対応するブロックが存在するか否かを調べる（Ｓ６０６）。

【００４１】その結果、命令キャッシュに当該ブロックが存在しなければ（Ｓ６０６：ＮＯ）、続けて、データキャッシュの当該ブロックがダーティであるか否かを調べて、ダーティであれば主記憶に当該ブロックを書き戻す（Ｓ６０７）。次に、書き込み対象アドレスに対応するブロックを主記憶から命令キャッシュへ読み出す（Ｓ６０８）。これにより、主記憶、データキャッシュ、命令キャッシュの内容が一致するので、データキャッシュと命令キャッシュに書き込み対象の命令を書き込み（Ｓ６０９）、処理を終了する（Ｓ６１０）。

【００４２】一方、命令キャッシュに当該ブロックが存在すれば（Ｓ６０６：ＹＥＳ）、データキャッシュと命令キャッシュの両方にブロックが存在することになるので、データキャッシュと命令キャッシュに書き込み対象の命令を書き込み（Ｓ６０９）、処理を終了する（Ｓ６１０）。

【００４３】なお、上述した処理フローでは、データキャッシュに書き込み対象アドレスに対応するブロックが存在しなければ、主記憶からデータキャッシュへの読み出しを行っているが（Ｓ６０５）、命令キャッシュとデータキャッシュの間でデータの転送が可能であれば、命令キャッシュに対応するデータが存在する場合は、主記憶との間の転送は省略することができる。同様に、書き込み対象アドレスに対応するブロックを主記憶から命令キャッシュへ読み出しているが（Ｓ６０８）、命令キャッシュとデータキャッシュの間でデータの転送が可能で、データキャッシュに対応するデータが存在する場合は、主記憶との間の転送は省略し、命令キャッシュからコピーするように最適化することができる。

【００４４】次に、あるデータ書き込みがデータキャッシュを介した書き込みか、命令キャッシュを介した書き込みかを指定・判別する方法について説明する。このような方法として、例えば、以下の（ａ）～（ｃ）に示すようなものが考えられる。

【００４５】（ａ）ページなど所定のメモリ領域毎に、命令キャッシュ及びデータキャッシュのいずれを介して書き込みを行う領域かを指定できるようにし、書き込み対象のアドレスによって判別する方法。

【００４６】（ｂ）ＣＰＵに内部フラグを設け、フラグの設定によって、命令キャッシュ及びデータキャッシュのいずれを介して書き込みを行うモードであるかを指定・判別する方法。

【００４７】（ｃ）データ書き込み命令として、命令キャッシュを介したものと、データキャッシュを介したものとを分けて用意し、これらの命令を使い分けることで、指定・判別する方法。

【００４８】まず、上述した（ａ）の方法に対応するものとして、マイクロプロセッサの命令セットのなかに、メモリ領域毎に、データキャッシュを介して書き込みが行われる領域か、命令キャッシュを介して書き込みが行われる領域かを指定するための命令（例えば、「set-dwrite」及び「set-iwrite」）を設けた場合について説明する。プロセッサは、メモリに対する書き込みを実行する際、書き込み対象アドレスがいずれの領域に該当するのか調べて、その結果に応じて、命令キャッシュまたはデータキャッシュを介した書き込みを行う。

【００４９】命令「set-dwrite」は、そのオペランドによって指定されるアドレス領域（例えば、オペランドで指定されるアドレスから１ページ分）は、データキャッシュを介して書き込みが行われるべき領域であることを

指定する命令であり、当該命令実行後は、指定されたアドレス領域に対する書き込みはデータキャッシュを介して行われるようになる。同様に、命令「set-iwrite」は、オペランドによって指定されたアドレス領域は、命令キャッシュを介して書き込みが行われるべき領域であることを指定する命令であり、当該命令実行後は、指定されたアドレス領域に対する書き込みは命令キャッシュを介して行われる。

【００５０】図６は、命令「set-dwrite」及び「set-iwrite」を設けた場合のハードウェア構成例を示す図である。これらの命令は、命令フェッチ・デコードユニット１００１によりメモリからの読み出され、デコードされて、実行制御が行われる。命令「set-dwrite」又は命令「set-iwrite」が実行されると、命令キャッシュを介した書き込みかデータキャッシュを介した書き込みかを示すビットとともに、オペランドで指定されたアドレスが、方向テーブル１００２に記憶される。

【００５１】そして、メモリへの書き込みを行う命令、例えば、ストア命令が実行され、データのストアを行う場合には、まず、ストア命令で指定されたアドレッシング・モードに従って、レジスタ１００３の値とストア命令中で指定された即値（immediate）のいずれかをマルチプレクサ１００４によって選択し、加算器１００５で、レジスタ１００３の値と演算して、データを格納する実効アドレスを生成する。そして、生成されたアドレスが、方向テーブル１００２に存在するか否かを比較器１００６により確認し、同時にそのアドレスに対して指定された書き込み対象キャッシュが命令キャッシュであるか否かを比較器１００７で調べる。その結果、方向テーブル１００２にストア・アドレスが存在し、かつ書き込み対象キャッシュが命令キャッシュであると指定されていた場合は、デマルチプレクサ１００８及びデマルチプレクサ１００９によって、書き込みアドレス、書き込みデータの供給先として、命令キャッシュ１０１１を選択し、書き込みを行う。一方、方向テーブル１００２にストア・アドレスが存在しないか、又は、書き込み対象キャッシュが命令キャッシュであると指定されていない場合は、デマルチプレクサ１００８及びデマルチプレクサ１００９によって、書き込みアドレス、書き込みデータの供給先として、データキャッシュ１０１０を選択し、書き込みを行う。なお、比較器１００６が比較対象とするビット数は、指定される領域の大きさによって、定まる。

【００５２】次に、上述した（ｂ）の方法に対応するものとして、マイクロプロセッサの内部に、メモリに対する書き込みを、データキャッシュを介して行うモードであるか、命令キャッシュを介して行うモードであるかを表すフラグを設けるとともに、命令セットのなかにフラグ設定命令（例えば、「set-iwrite-flag」及び「reset-iwrite-flag」）を設けた場合について説明する。命令「set-iwrite-flag」及び「reset-iwrite-flag」は、ＣＰＵに設けられた内部フラグ（iwrite-flag）を制御（セット／リセット）するための命令であり、ＣＰＵは、このフラグの設定値に従って、命令キャッシュを介してデータの書き込みを行うか、データキャッシュを介してデータの書き込みを行うかを決定する。

【００５３】ここで、命令「set-iwrite-flag」は、命令キャッシュを介して書き込みが行われるように、内部フラグをセットする命令であり、当該命令実行後の書き込み命令はすべて命令キャッシュを介して行われる。これに対して、命令「reset-iwrite-flag」は、データキャッシュを介して書き込みが行われるように、内部フラグをリセットする命令であり、当該命令実行後の書き込みはすべてデータキャッシュを介して行われる。

【００５４】図７は、内部フラグ並びに命令「set-iwrite-flag」及び「reset-iwrite-flag」を設けた場合のハードウェア構成例を示す図である。命令「set-iwrite-flag」及び「reset-iwrite-flag」は、命令フェッチ・デコードユニット１１０１により、メモリから読み出され、デコードされ、実行制御が行われる。命令「set-dwrite-flag」又は「set-iwrite-flag」が実行されると、現在の書き込みモードを表すフラグが内部フラグ（iwrite-flag）１１０２に記憶される。

【００５５】そして、ストア命令等によりデータの書き込みを行う場合には、図６の場合と同様にして実効アドレスを、レジスタ１１０３、マルチプレクサ１１０４、加算器１１０５によって生成する。そして、デマルチプレクサ１１０６、１１０７では、フラグ１１０２の値に従ってアドレスおよびデータの供給先を選択し、データキャッシュ１１０９または命令キャッシュ１１０８へ書き込みを行う。

【００５６】最後に、上述した（ｃ）の方法に対応するものとして、命令コードによって命令キャッシュを介してデータの書き込みを行うか、データキャッシュを介してデータの書き込みを行うかを指定・決定する場合について説明する。すなわち、マイクロプロセッサの命令セットのなかに、データ書き込み命令として、通常の、データキャッシュを介してデータの書き込みを行う命令（例えば、「write」）の他に、命令キャッシュを介してデータの書き込みを行う命令（例えば、「iwrite」）を用意し、ソフトウェア等は、これら２種の命令を適宜選択して使用することにより、主記憶への書き込みの際に介するキャッシュの選択を行う。

【００５７】命令「iwrite」が実行されると、オペランドで指定された実効アドレスに対して、オペランドで指定したレジスタ等に格納されたデータ（命令）を、命令キャッシュを介して書き込む。

【００５８】図８は、命令「iwrite」を設けた場合のハードウェア構成例を示す図である。命令「iwrite」は、命令フェッチ・デコードユニット１２０１により主記憶

や命令キャッシュからの読み出され、デコードされ、実行制御が行われる。命令キャッシュを対象としたストア命令「iwrite」により、データのストアを行う場合には、まず、前述したのと同様にして、レジスタ１２０３、マルチプレクサ１２０４、加算器１２０５でアドレスを生成する。デマルチプレクサ１２０６、１２０７では、命令フェッチ・デコードユニット１２０１からの、当該命令が命令キャッシュを対象としたストア命令であることを表す出力信号１２０２によりデータの書き込み先として、命令キャッシュ１２０８を選択し、書き込みを行う。

【００５９】同様にデータキャッシュを対象としたストア命令「write」によりデータのストアを行う場合には、デマルチプレクサ１２０６、１２０７では命令フェッチ・デコードユニット１２０１からの出力信号１２０２によりアドレスおよびデータの供給先として、データキャッシュを選択し、データキャッシュ１２０９への書き込みを行う。

【００６０】以上説明したような、命令キャッシュへの書き込みをソフトウェア的に制御できる機構をマイクロプロセッサに設けることにより、主記憶とキャッシュ間（又は、上位キャッシュと下位キャッシュ間）の不要なデータの転送を減少させ、動的な命令生成を行うプログラムの性能を向上させることが可能となる。

【００６１】最後に、上述したような本発明によるマイクロプロセッサ上で実行される動的コンパイラについて説明する。本発明によるマイクロプロセッサでは、命令キャッシュへの書き込みをソフトウェア的に制御できるので、動的コンパイラは、プログラム実行時に動的に生成した命令を主記憶に格納する際、上述したいずれかの方法を使って、命令キャッシュを介して書き込むようにする。これ以外の動作は、従来の動的コンパイラの動作と同様でよい。このようにすれば、動的に生成した命令を主記憶に格納する場合の主記憶参照の必要性が減り、プログラムの高速実行が可能になる。

【００６２】

【発明の効果】以上、詳細に説明したように、本発明によれば、実行時に動的に機械語命令の生成を行うプログラムにおいて、生成した機械語命令を直接命令キャッシュへ書き込むことが可能となる。これにより、実行時に動的に機械語命令の生成を行うプログラムの実行の高速化が図れる。

【図面の簡単な説明】

【図１】　本発明による非統合型キャッシュのマイクロプロセッサの動作概要を説明する図である。

【図２】　命令キャッシュから主記憶への書き込みパスを設けたマイクロプロセッサにおける、動的に生成された命令の流れを示す図である。

【図３】　命令キャッシュから主記憶への書き込みパスを設けたマイクロプロセッサにおける、書き込み処理の流れを示す図である。

【図４】　命令キャッシュから主記憶への書き込みパスを有しないマイクロプロセッサにおける、動的に生成された命令の流れを示す図である。

【図５】　命令キャッシュから主記憶への書き込みパスを有しないマイクロプロセッサにおける、書き込み処理の流れを示す図である。

【図６】　所定のメモリ領域毎に、書き込み対象キャッシュを指定できる命令を設けたマイクロプロセッサのハードウェア構成例を示す図である。

【図７】　命令キャッシュを介して書き込むモードか、データキャッシュを介して書き込むモードかを指定する内部フラグを設けたマイクロプロセッサのハードウェア構成例を示す図である。

【図８】　命令キャッシュを介して書き込みを行う命令を設けたマイクロプロセッサのハードウェア構成例を示す図である。

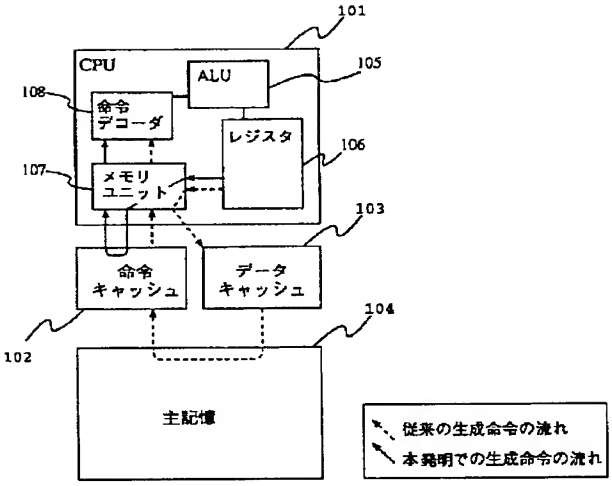
【図９】　本発明が適用される計算機システムの例を示す図である。

【符号の説明】

１０１　ＣＰＵ
１０２　命令キャッシュ
１０３　データキャッシュ
１０４　主記憶
１０５　ＡＬＵ
１０６　レジスタ
１０７　メモリユニット
１０８　命令デコーダ

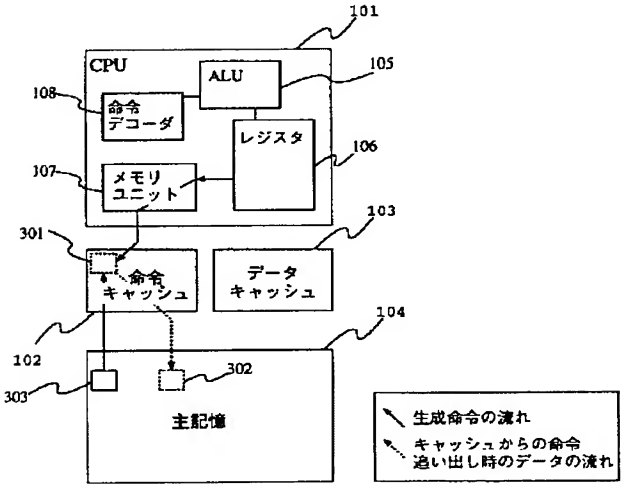
【図１】

図1



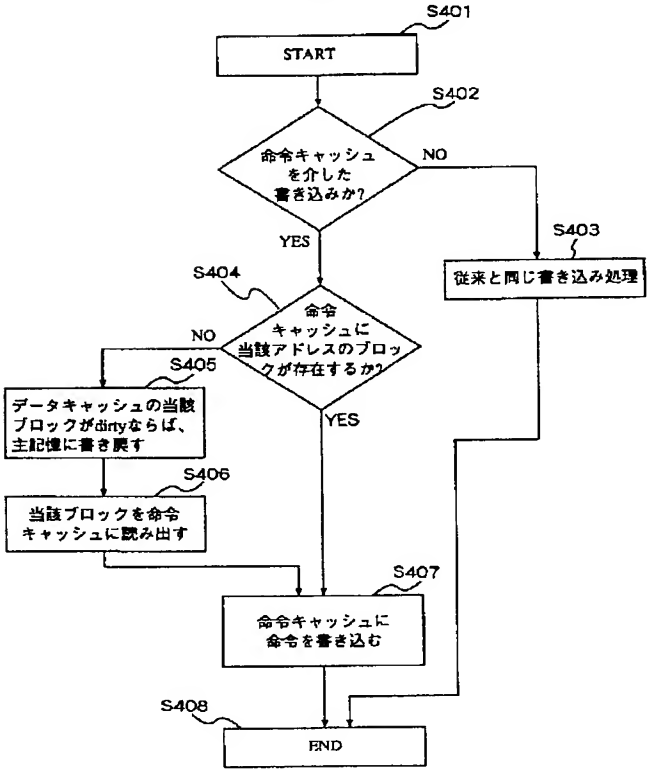
【図２】

図2



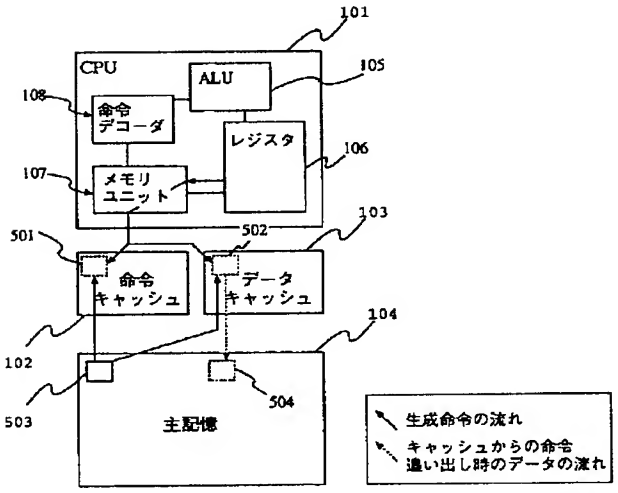
【図３】

図3



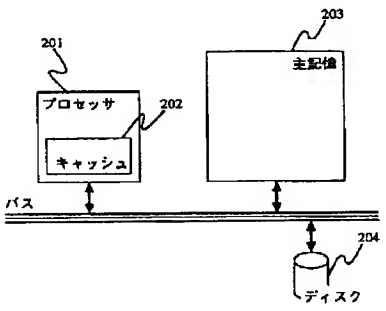
【図４】

図4



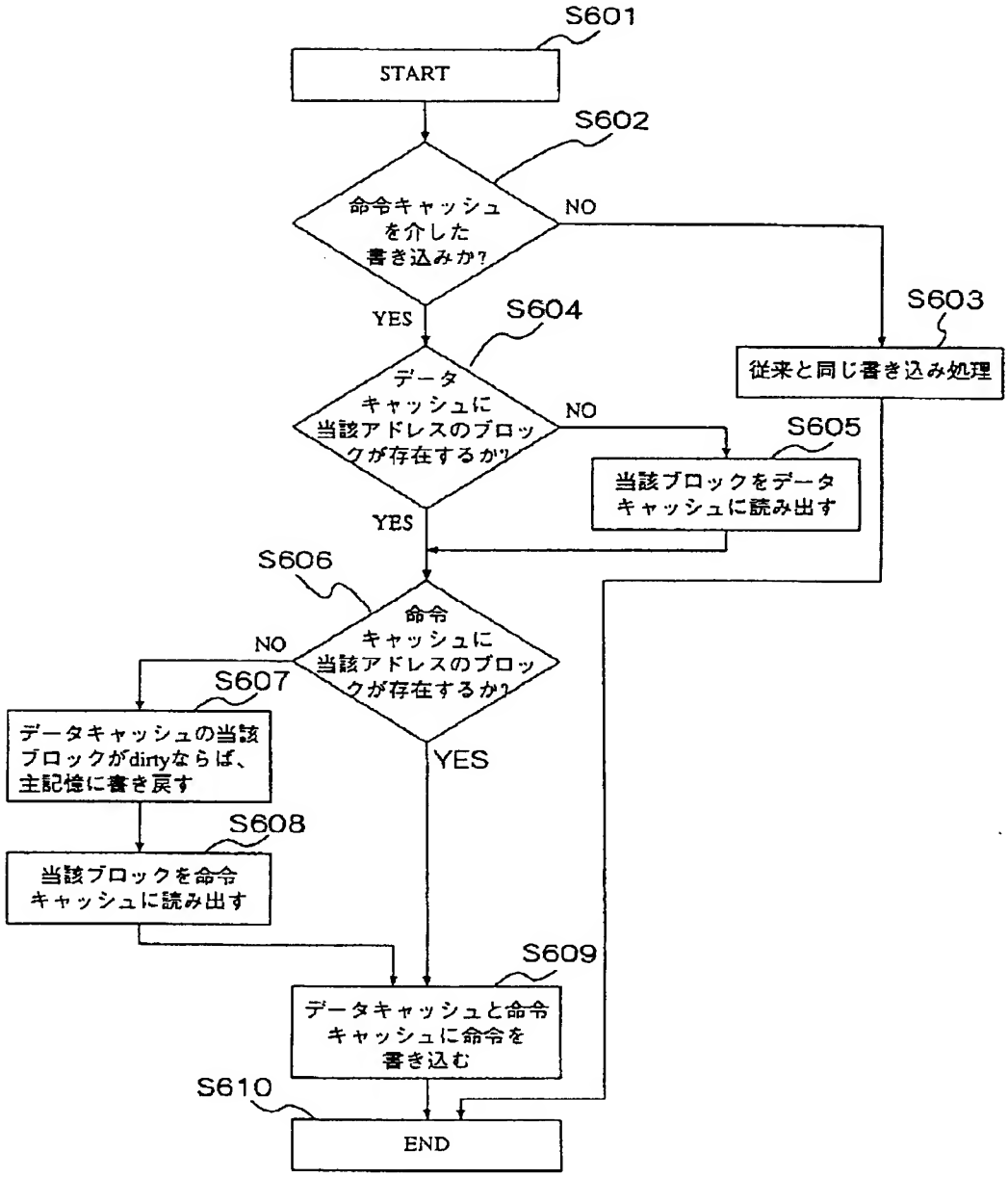
【図９】

図9



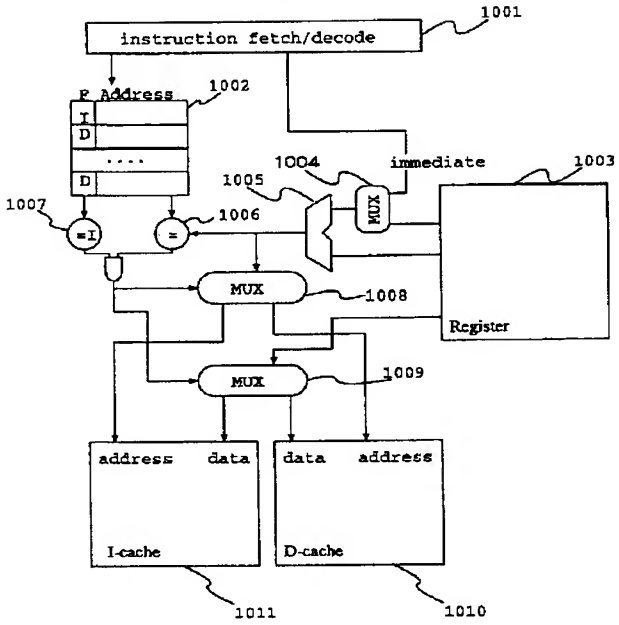
【図５】

図5


S601
START
S602
命令キャッシュを介した書き込みか?
NO
YES
S603
従来と同じ書き込み処理
S604
データキャッシュに当該アドレスのブロックが存在するか?
NO
S605
当該ブロックをデータキャッシュに読み出す
YES
S606
命令キャッシュに当該アドレスのブロックが存在するか?
NO
S607
データキャッシュの当該ブロックがdirtyならば、主記憶に書き戻す
YES
S608
当該ブロックを命令キャッシュに読み出す
S609
データキャッシュと命令キャッシュに命令を書き込む
S610
END

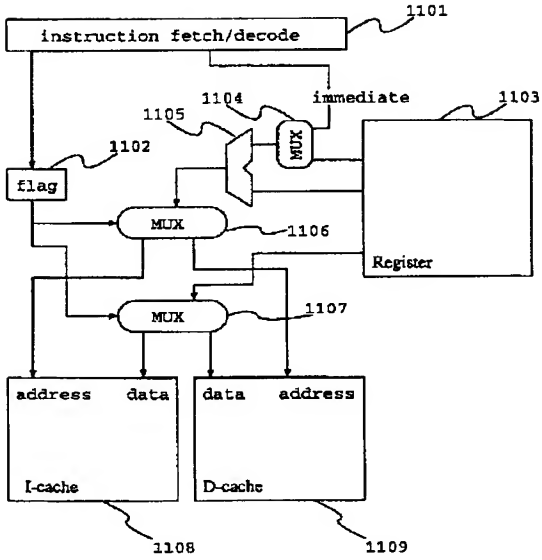
【図６】

図6



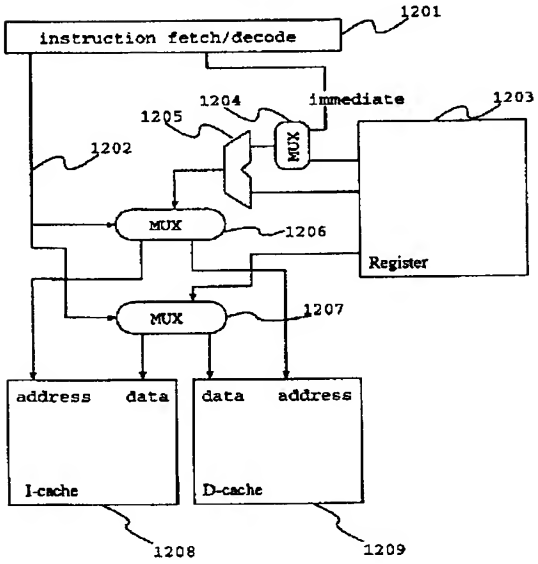
【図７】

図7



【図８】

図8



* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]
[Claim 1]A processor having the command which can specify a field which writes in into an instruction set via a field where the field concerned writes in via a data cache for every predetermined memory area, and an instruction cache.
[Claim 2]The processor according to claim 1 characterized by writing in via an instruction cache when writing is directed to a memory area specified as a field which writes in via an instruction cache with said command.
[Claim 3]It has a flag showing whether it is the mode which writes in via a data cache, or it is the mode which writes in via an instruction cache, A processor characterized by writing in via an instruction cache when it is shown that it is the mode in which the flag concerned writes in via an instruction cache.
[Claim 4]The processor according to claim 3 having the command which operates said flag in an instruction set.
[Claim 5]A processor having the command which directs to perform writing to a memory via an instruction cache into an instruction set.
[Claim 6]The processor according to any one of claims 1 to 5 characterized by writing in both an instruction cache and a data cache when writing through an instruction cache is directed.
[Claim 7]A method of dynamic compile characterized by making it write in via an instruction cache when writing in a memory a command which is the method of dynamic compile using the processor according to any one of claims 1 to 6, and was generated dynamically.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Field of the Invention]This invention relates to a microprocessor suitable for performing the virtual machine interpreter etc. which perform especially by generating a machine language instruction program dynamically at the time of execution about a processor.
[0002]
[Description of the Prior Art]In the conventional microprocessor, cash is formed as a high-speed intermediate storage between main memory (main memory) and CPU in many cases. This cash can be classified into the unified type cash which unified that disintegrate type cash that separated the data cache and the instruction cache constitutionally, and a data cache and an instruction cache. Since the former disintegrate type cash can give an independently high data supply bandwidth to each of a data cache and an instruction cache as compared with the latter unified type cash, generally it can obtain high performance. For this reason, in a high-end microprocessor, the cash of a disintegrate type is used in many cases.
[0003]Things, such as Java (Java is a trademark of U.S. Sun Microsystems and Inc.) and Smalltalk, are known as a programming language which describes the program executed by such a microprocessor on the other hand. The program described with these programming languages is once changed into the command of the virtual computer called a byte code, and a byte code is usually executed by the software called a virtual machine interpreter.
[0004]However, the execution by such a virtual machine interpreter has large overheads, such as interpretation execution of a virtual machine command. Therefore, generally compared with the case where the program described by Java or Smalltalk is described with the conventional compiler languages, such as C and FORTRAN, the execution speed becomes a low speed. On the other hand, for example, L.P. . it is based on Deutsch and A.M. Chiffman . "Efficient Implementation. of the Smalltalk-80. System." As shown in (In Proceedings of the 11th Annual ACM Symposium on Principles of Programming Languages pp.297-302, 1984), The technique of accelerating interpreter execution by generating the machine language instruction which performs processing equivalent to it at the time of execution, and carrying out immediate execution of the generated machine language instruction to the virtual machine instruction sequence for execution, is used. Dynamic compile since such a technique generates a machine language instruction dynamically at the time of execution, Or since it compiles when trying to perform, it is called JIT (just-in-time: Just-In-Time) compile, and it is used for improvement in the speed of the Java virtual machine, etc.
[0005]
[Problem(s) to be Solved by the Invention]Although the JIT compile art which generates a command dynamically at the time of such execution is effective in improvement in the speed of a virtual machine interpreter etc., Since it is necessary to write the command generated at the time of execution in main memory at the time of execution when using by the microprocessor which adopted the cache mechanism of the disintegrate type is considered, the following problems will arise.
[0006]Since the writing to the main memory of the command generated dynamically is not

different from the writing of the usual data when it sees from a processor, it will be performed via a data cache by the processor which adopted disintegrate type cash. For this reason, since the compatibility of an instruction cache and a data cache is not maintained even if it is going to execute the command concerned written in immediately after writing the generated command in a memory via a data cache, it cannot perform as it is. That is, since renewal of an instruction cache is not performed, an unexpected command may be executed as it is. So, in almost all microprocessors, it has the command for returning the data on the specified data cache to main memory, or cancelling the cache block on an instruction cache. And before executing the command written in main memory, while canceling the data on the instruction cache corresponding to the written-in address using such a command (cancellation), If it is write back (write-back) type cash, the data on the data cache corresponding to the written-in address will be returned to main memory so that change may be reflected in main memory.

[0007]That is, dynamic generation and executive operation of a command are performed by the following flows.

[0008]1. Cache block of data cache corresponding to write-in object address (henceforth address A), Write-in 2. of the machine language instruction generated dynamically Cancellation 3. of the cache block of the instruction cache corresponding to the address A Write return to the main memory of the cache block of the data cache corresponding to the address A (in the case of write back type cash)

4. As read-out of the command from the read-out 5. instruction cache to the cache block to which an instruction cache corresponds from the address A of main memory, and beyond execution stated, If a command tends to be dynamically generated at the time of execution and it is going to execute the generated command concerned in the microprocessor which adopted disintegrate type cash, the command generated dynamically, Since it is written out to main memory and it is necessary to read to an instruction cache after that after being written in a data cache, it will be accompanied by the writing and read-out to main memory, and this can become a fall factor of execution speed.

[0009]In the high-end microprocessor which can process many commands in time which the degree of command level parallel and clock frequency within a processor improve, and referring to the main memory takes especially. There is a possibility that the problem that the performance of a program of operation in which the overhead which referring to the main memory takes performs dynamic command generation processing will fall may aggravate.

[0010]In the conventional static compiler languages, such as C or FORTRAN, in order that conversion to a machine language instruction might be beforehand performed before program execution and might not generate a command dynamically at the time of execution, even if it was the cache constitution of the discrete type, it did not become a big problem. In the processing which generates a machine language instruction dynamically at the time of execution like the JIT compile in a virtual machine interpreter, the purpose of this invention is to provide the processor which can press down low the degradation produced by refer to the main memory of a dynamic command generate time.

[0011]

[Means for Solving the Problem]A processor concerning this invention has the command which can specify a field which writes in into an instruction set via a field where the field concerned writes in via a data cache, and an instruction cache for every predetermined memory area. In this case, when writing is directed to a memory area specified as a field which writes in via an instruction cache with said command, it writes in via an instruction cache.

[0012]In the mode in which the 2nd processor concerning this invention writes in via a data cache. It has a flag showing whether it is the mode which writes in via an instruction cache, and when it is shown that it is the mode in which the flag concerned writes in via an instruction cache, it writes in via an instruction cache. In this case, it is preferred to have the command which operates said flag in an instruction set.

[0013]It has the command which directs that the 3rd processor concerning this invention performs writing to a memory via an instruction cache into an instruction set.

[0014]When writing which passed an instruction cache in the 3rd processor from the above 1st

is directed, it may be made to write in both an instruction cache and a data cache. Since a copy of update information of an instruction cache will exist in a data cache if it does in this way, Write return to main memory of update information when [from an instruction cache to a memory / direct] cash overflow arises in an instruction cache, without having written and establishing a return course becomes possible.

[0015]A method of dynamic compile concerning this invention is written in via an instruction cache, when writing in a memory a command which is the method of dynamic compile using the 1st to 3rd above-mentioned processor, and was generated dynamically.

[0016]In a processor by this invention, since writing to a memory can be performed via an instruction cache, When a command generated dynamically is written in a memory by dynamic compile, It can write in via an instruction cache, the necessity for referring to the main memory can decrease, and degradation produced by refer to the main memory of a dynamic command generate time can be low pressed down in processing which generates a machine language instruction dynamically at the time of execution like JIT compile in a virtual machine interpreter.

[0017]

[Embodiment of the Invention]Hereafter, an embodiment of the invention is described in detail, referring to drawings.

[0018]Drawing 9 is a figure showing the example of the computer system which carries out this invention. The software which performs command generation processing at the time of execution, such as a JIT compiler in a virtual machine interpreter, is read from the disk unit 204 to the main memory 203, and is performed by the processor 201. The machine language instruction generated by this software is written between the main memory 203 and the processor 201 via the cash 202 which is a high-speed intermediate storage. Since it is easy, the memory hierarchy is made into cash and two hierarchies of main memory here, but this invention can also be applied to a microprocessor with two or more cache hierarchies.

[0019]Drawing 1 is a figure explaining the operation outline of the microprocessor which adopted the disintegrate type cash by this invention. In order to contrast with this invention, operation of the conventional microprocessor which adopted disintegrate type cash is first explained using the figures. In the conventional microprocessor which adopted disintegrate type cash, the command which CPU101 executes is read from the main memory 104 to the instruction cache 102, is received to CPU101 and passed. Thus, the fetched command is passed to the instruction decoder 108 via the memory unit 107. The command concerned is decoded by the instruction decoder 108, ALU105 and the register 106 are controlled based on the decoded result concerned, and execution of a command is performed. Since it becomes unnecessary to refer to the low speed main memory 104 by saving the command executed frequently at the instruction cache 102 which is a high-speed intermediate storage, it becomes possible to perform high-speed execution.

[0020]In drawing 1, a dashed line expresses the flow of the command by dynamic command generation processing generated dynamically in the conventional microprocessor. The solid line expresses the flow of the command in this invention generated dynamically. In explanation here, since it is easy, the case where overflow by competition or capacity lacks of a cache block does not arise is explained first, and the case where overflow of a cache block occurs is mentioned later.

[0021]As the dashed line of the figure shows, after the command dynamically generated by ALU105 at the time of execution is stored in the register 106, in the conventional dynamic command generation processing, it is written in the data cache 103 via the memory unit 107. When the data cache 103 has adopted the write back method, since the command which should be executed exists only on the data cache 103, in this state, the cache block in which the command concerned was written is returned to the main memory 104. Since the cache block corresponding to the address which performed the writing concerned on the instruction cache 102 may exist, in existing, it cancels the block concerned on the instruction cache 102. The cache block corresponding to the address which wrote in the command generated dynamically by this, Since it stops existing on the instruction cache 102, when it is going to execute the generated command, A command is read from the main memory 104 on the instruction cache

102, and via the memory unit 107, popularity is won to the instruction decoder 108, it is passed, and the instruction execution of CPU101 is performed according to a decoded result.
[0022]On the other hand, in the flow of the generation instruction in the dynamic command generation processing in this invention, after the command which ALU105 generated dynamically is stored in the register 106, it is directly written in the instruction cache 102 via the memory unit 107. And since the command generated on the instruction cache 102 exists in executing the generated command, the memory unit 107 reads the command which should be executed from the instruction cache 102, and passes the read command to the instruction decoder 108.
[0023]As explained above, in the conventional microprocessor. It did not have to be concerned with the existence of the margin of cache capacity, but the command written in the data cache 103 had to be written in the low speed main memory 104 from once high-speed cache memory, and it had to read to CPU101 via the instruction cache 102 again. On the other hand, in the microprocessor by this invention, since the command written in on the direct instruction cash 102 via the main memory 104 can be executed, execution of the command generated dynamically is accelerable.
[0024]In the above explanation, neither competition of a cache block nor dismissal of the cache block by capacity lacks was taken into consideration. Actually, cache capacity is restricted, and since it produces, the competition of a cache block needs to take into consideration the processing at the time of dismissal. So, below, the command writing processing in consideration of the processing at the time of cash dismissal is explained.
[0025]In the writing processing to cash, the processings differ by whether the processing in the case of driving out a cache block by competition etc. has adopted whether cash has adopted the write-through (write-through) method and a write back method.
[0026]What is necessary is just in the case of write through system, to only cancel the cache block for dismissal, since the writing to main memory is also performed simultaneously with the writing to cash. Namely, when the instruction cache by this invention is constituted as cash of write through system, The newest data will be saved on main memory and what is necessary is just to cancel the contents of the cache block at the time of dismissal of a cache block, since the command for writing is written in both an instruction cache and main memory.
[0027]On the other hand, since the newest data exists only on cash, in the case of a write back method, the data to main memory needs to write at the time of dismissal of the cache block containing update information, and it is necessary to it to carry out return processing. In the conventional processor, since the writing of the data from CPU to cash was performed only to the data cache, the data from cash to main memory wrote, and return processing was performed only in the data cache. In the processor by this invention, since the writing to an instruction cache is also performed, when a write back method is adopted, the data to the main memory from an instruction cache needs to write, and it is necessary to take return processing into consideration. The method which the data to the main memory from such an instruction cache writes, and enables the writing from an instruction cache to direct main memory at return processing, and the method the writing to main memory is made to hold only from a data cache as usual can be considered. Below, the example of realization is shown about an all directions type.
[0028]Drawing 2 expresses the flow of the command generated dynamically at the time of extending the architecture so that a cache block can be written in from an instruction cache to main memory. In drawing 2, the command which was dynamically generated by ALU105 and was stored in the register 106 is written in the instruction cache 102 via the memory unit 107. Here, when the command corresponding to an address which is different from a writing address in the cache block 301 used as a write-in object is already held, it is necessary to make the cache block available but, and the processing changes with updating states of the cache block 301 concerned. Since the contents on the main memory 104 and the contents on the instruction cache 102 are in agreement if the cache block 301 is in the state (clean : clean) where writing is not performed, Write return to the main memory 104 is not performed, but cancels the data of the cache block 301 on the instruction cache 102. On the other hand, since the newest command exists only on the instruction cache 102 if the cache block 301 is in the state (dirty :

dirty) where writing was performed, the cache block 301 is first returned to the field 302 on corresponding main memory.
[0029]After cancellation or write return of the original data on the instruction cache 102 is completed, the data block 303 on the main memory 104 corresponding to the write-in object address of the command generated dynamically is read into the cache block 301, The command for writing which received from the memory unit 107 and was passed is written in. These operations are the same as operation of the usual write back type data cache.
[0030]Drawing 3 is a figure showing the flow of the writing processing to the main memory at the time of the cache block from the instruction cache shown in drawing 2 to main memory writing, and adding a return mechanism. If processing is started as shown in the figure (S401), writing will distinguish first whether it is the writing through an instruction cache (S402). The writing for which a certain data writing instruction passed the data cache, and the writing through an instruction cache is later mentioned about the method of specifying and judging.
[0031]As a result of distinction, the writing concerned performs writing processing which passed the same data cache as the former via the data cache (S402:NO) (S403), and ends processing (S408). On the other hand, if it is the writing through an instruction cache (S402:YES), it will be investigated whether the cache block corresponding to a write-in object address exists on an instruction cache (S404). When the cache block corresponding to a write-in object address does not exist on an instruction cache, as a result, (S404:NO), It is investigated whether continuously, it writes in a data cache and the cache block corresponding to an object address exists, Such a cache block exists, if the cache block concerned of a data cache is in a dirty state, the block concerned will be written out to main memory (S405), and the block corresponding to the address for writing is read to an instruction cache (S406). Since it writes in on an instruction cache and the cache block of an object address is secured by this, the command dynamically generated by the block concerned of the instruction cache is written in (S407), and processing is finished (S408). On the other hand, since it writes in on (S404:YES) and an instruction cache and the cache block of the object address is already secured when the cache block corresponding to a write-in object address exists on an instruction cache, Then, a command is written in an instruction cache (S407), and processing is finished (S408).
[0032]When the cache block corresponding to a command write-in object address exists on a data cache, What is necessary is to cancel the data block of an applicable data cache (cancellation), or just to write a command also in a data cache with an instruction cache, in order to maintain compatibility with an instruction cache.
[0033]Drawing 4 expresses the flow of the command generated dynamically when not having a mechanism which writes in from the instruction cache 102 directly to the main memory 104. In this case, compared with the direct thing from an instruction cache to main memory shown in drawing 2 since it did not need to write and a return course did not need to be established, there are few hardware changing amounts to the conventional microprocessor, and they end.
[0034]As shown in drawing 4, the command which was dynamically generated by ALU105 and was stored in the register 106 is written in the instruction cache 102 and the data cache 103 via the memory unit 107. That is, suppose that it has a copy of the command concerned also on the data cache 103.
[0035]Thus, having a copy of the command (corresponding cache block) concerned also on the data cache 103, Since the cache block from the instruction cache 102 to the main memory 104 writes and a return path does not exist in this embodiment, When writing in the command dynamically generated to the instruction cache 102, and cash overflow arose and the write return to main memory is needed, it is because it can be made to perform the write return concerned.
[0036]When performing the writing to the instruction cache 102, and cash overflow arose and the write return to main memory is needed, While canceling the block 501 on the instruction cache which wrote and became a return object, the update state of the block 502 on the corresponding data cache 103 is investigated, and if dirty, it will return to the field 504 on the corresponding main memory 104. Next, the block 503 corresponding to a write-in object address is read from the main memory 104, and it reads into both the instruction cache 102 and the data cache 103.

And the command for writing is written in the block 501 on the instruction cache 102, and the block 502 on the data cache 103.
[0037]Although the cache block secured on the data cache 103 corresponding to the data on the instruction cache 102 may be returned to main memory by competition by the writing of the data to a data cache, In that case, since the contents of the cache block on the instruction cache 102 and the data on the main memory 104 are in agreement, it is not necessary to perform special operation.
[0038]Drawing 5 is a figure showing the flow of the writing processing to the main memory when the cache block from the instruction cache shown in drawing 4 to main memory writing, and not adding a return mechanism. As shown in the figure, when performing the writing to main memory (S601), it checks first that it is the writing through an instruction cache (S602). Writing mentions the specification / judgment method of being the writing through an instruction cache later.
[0039]If writing is not writing through an instruction cache as a result of a check (S602:NO), writing processing through the usual data cache will be performed (S603), and writing processing will be ended (S610).
[0040]On the other hand, if it is the writing through an instruction cache (S602:YES), it will be investigated whether first, it writes in a data cache and the block corresponding to an object address exists (S604). As a result, if a corresponding block does not exist in a data cache (S604:NO), the block concerned is read to a data cache (S605). It is investigated whether it writes in an instruction cache and the block corresponding to an object address exists (S606).
[0041]As a result, if the block concerned does not exist in an instruction cache (S606:NO), it investigates continuously whether the block concerned of a data cache is dirty, and if dirty, the block concerned will be returned to main memory (S607). Next, the block corresponding to a write-in object address is read from main memory to an instruction cache (S608). Thereby, since the contents of main memory, a data cache, and the instruction cache are in agreement, it writes in a data cache and an instruction cache, the target command is written in (S609), and processing is ended (S610).
[0042]Since a block will exist in both a data cache and an instruction cache on the other hand if the block concerned exists in an instruction cache (S606:YES), it writes in a data cache and an instruction cache, the target command is written in (S609), and processing is ended (S610).
[0043]If it writes in a data cache and the block corresponding to an object address does not exist in the process flow mentioned above, Although read-out to a data cache from main memory is performed (S605), if transmission of data is possible between an instruction cache and a data cache, when the data corresponding to an instruction cache exists, the transmission between main memory can be omitted. Similarly, although the block corresponding to a write-in object address is read from main memory to the instruction cache (S608), a data transfer is possible between an instruction cache and a data cache, When the data corresponding to a data cache exists, the transmission between main memory can be omitted, and it can be optimized so that it may copy from an instruction cache.
[0044]Next, a certain data writing explains how to specify and distinguish the writing through a data cache, and the writing through an instruction cache. As such a method, a thing as shown in the following (a) - (c) can be considered, for example.
[0045](a) How to enable it to specify whether it is a field which writes in via any of an instruction cache and a data cache for every predetermined memory areas, such as a page, and distinguish with the address for writing.
[0046](b) How to specify and distinguish whether it is the mode which forms an internal flag in CPU and writes in via any of an instruction cache and a data cache by setting out of a flag.
[0047](c) How to specify and distinguish by dividing and preparing the thing through an instruction cache, and the thing through a data cache as a data writing instruction, and using these commands properly.
[0048]First, as a thing corresponding to the method of (a) mentioned above in the instruction set of a microprocessor, The case where the command (for example, "set-dwrite" and "set-iwrite") for specifying the field where writing is performed via a data cache, and the field where writing is performed via an instruction cache for every memory area is provided is explained. When a

processor performs the writing to a memory, it investigates whether a write-in object address corresponds to which field, and performs the writing through an instruction cache or a data cache according to the result.

[0049]The address area (from the address specified with an operand to for example, 1 page) specified by the operand a command "set-dwrite", It is the command which specifies that it is a field where writing should be performed via a data cache, and the writing to the address area where after the instruction execution concerned was specified comes to be performed via a data cache. Similarly the address area specified by the operand a command "set-iwrite", It is the command which specifies that it is a field where writing should be performed via an instruction cache, and the writing to the address area where after the instruction execution concerned was specified is performed via an instruction cache.

[0050]Drawing 6 is a figure showing the example of hardware constitutions at the time of providing a command "set-dwrite" and "set-iwrite." These commands are read from a memory with the instruction fetch decoding unit 1001, are decoded, and execution control is performed. If a command "set-dwrite" or a command "set-iwrite" is executed, the address specified with the operand with the bit which shows the writing through an instruction cache or the writing through a data cache will be memorized by the direction table 1002.

[0051]And in executing the command which performs the writing to a memory, for example, store instruction, and storing data. First, by the multiplexer 1004, according to the addressing mode specified by store instruction, choose immediate (immediate) either which was specified in the value of the register 1003, and store instruction, and with the adding machine 1005. It calculates with the value of the register 1003 and the effective address which stores data is generated. And it checks by the comparator 1006 whether the generated address exists in the direction table 1002, and it is investigated by the comparator 1007 whether the cash for writing simultaneously specified to the address is an instruction cache. As a result, when specified that a store address exists in the direction table 1002, and the cash for writing is an instruction cache, By the demultiplexer 1008 and the demultiplexer 1009, it writes in by choosing the instruction cache 1011 as a supply destination of a writing address and write data. On the other hand. [whether a store address exists in the direction table 1002, and] Or when not specified that the cash for writing is an instruction cache, it writes in by choosing the data cache 1010 as a supply destination of a writing address and write data by the demultiplexer 1008 and the demultiplexer 1009. The number of bits which the comparator 1006 makes a comparison object becomes settled by the area size specified.

[0052]Next, as a thing corresponding to the method of (b) mentioned above inside a microprocessor, While forming the flag showing whether it is the mode in which the writing to a memory is performed via a data cache, or it is the mode in which it carries out via an instruction cache, The case where a flag setting instruction (for example, "set-iwrite-flag" and "reset-iwrite-flag") is provided into an instruction set is explained. A command "set-iwrite-flag" and "reset-iwrite-flag", It is the command for controlling the internal flag (iwrite-flag) formed in CPU (a set/reset), and CPU determines whether to write in data via an instruction cache, or write in data via a data cache according to the preset value of this flag.

[0053]Here, a command "set-iwrite-flag" is a command which sets an internal flag so that writing may be performed via an instruction cache, and the write instruction after the instruction execution concerned is altogether performed via an instruction cache. On the other hand, a command "reset-iwrite-flag" is a command which resets an internal flag so that writing may be performed via a data cache, and the writing after the instruction execution concerned is altogether performed via a data cache.

[0054]Drawing 7 is a figure showing the example of hardware constitutions at the time of providing an internal flag, a command "set-iwrite-flag", and "reset-iwrite-flag." With the instruction fetch decoding unit 1101, a command "set-iwrite-flag" and "reset-iwrite-flag" are read from a memory, and are decoded, and execution control is performed. If a command "set-dwrite-flag" or "set-iwrite-flag" is performed, the flag showing the present write mode will be memorized by the internal flag (iwrite-flag) 1102.

[0055]And in writing in data by store instruction etc., the register 1103, the multiplexer 1104, and

the adding machine 1105 generate an effective address like the case of drawing 6. And in the demultiplexers 1106 and 1107, the supply destination of an address and data is chosen according to the value of the flag 1102, and it writes in to the data cache 1109 or the instruction cache 1108.

[0056]The case where it is specified and determined whether to write in data via an instruction cache as a thing corresponding to the method of (c) mentioned above at the end by an instruction code or write in data via a data cache is explained. Namely, in the instruction set of a microprocessor as a data writing instruction, The command which writes in data via the usual data cache. Software chooses the cash passed in the case of the writing to main memory by using these two sorts of commands, choosing them suitably by preparing the command (for example, "iwrite") which writes in data other than (for example, "write") via an instruction cache.

[0057]Execution of a command "iwrite" will write in the data (command) stored in the specified register etc. with the operand to the effective address specified with the operand via an instruction cache.

[0058]Drawing 8 is a figure showing the example of hardware constitutions at the time of providing a command "iwrite." A command "iwrite" is read from main memory or an instruction cache with the instruction fetch decoding unit 1201, and is decoded, and execution control is performed. By the store instruction "iwrite" for an instruction cache, in storing data, it generates an address with the register 1203, the multiplexer 1204, and the adding machine 1205 the same with having mentioned above first. In the demultiplexers 1206 and 1207, the command concerned from the instruction fetch decoding unit 1201 writes in by choosing the instruction cache 1208 as a writing destination of data with the output signal 1202 showing being the store instruction for an instruction cache.

[0059]In storing data by the store instruction "write" for a data cache similarly, In the demultiplexers 1206 and 1207, a data cache is chosen as a supply destination of an address and data with the output signal 1202 from the instruction fetch decoding unit 1201, and the writing to the data cache 1209 is performed.

[0060]It is between main memory and cash (.) by forming in a microprocessor the mechanism which can control by software the writing to an instruction cache which was explained above. Or the unnecessary data transfer between a high order cache and low rank cash is decreased, and it becomes possible to raise the performance of a program of performing dynamic command generation.

[0061]The dynamic compiler performed at the end on the microprocessor by this invention which was mentioned above is explained. In the microprocessor by this invention, since the writing to an instruction cache is controllable by software, a dynamic compiler is written in via an instruction cache using one of the methods mentioned above, when the command dynamically generated at the time of program execution is stored in main memory. The operation of those other than this may be the same as operation of the conventional dynamic compiler. If it does in this way, the necessity for referring to the main memory in the case of storing in main memory the command generated dynamically will decrease, and the high speed execution of a program will become possible.

[0062]

[Effect of the Invention]As mentioned above, as explained in detail, according to this invention, in the program which generates a machine language instruction dynamically at the time of execution, it becomes possible to write the generated machine language instruction in direct instruction cash. Improvement in the speed of execution of the program which generates a machine language instruction dynamically by this at the time of execution can be attained.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]
[Drawing 1]It is a figure explaining the operation outline of the microprocessor of the disintegrate type cash by this invention.
[Drawing 2]It is a figure in the microprocessor which formed the write-in path from an instruction cache to main memory showing the flow of the command generated dynamically.
[Drawing 3]It is a figure in the microprocessor which formed the write-in path from an instruction cache to main memory showing the flow of writing processing.
[Drawing 4]It is a figure in the microprocessor which does not have a write-in path from an instruction cache to main memory showing the flow of the command generated dynamically.
[Drawing 5]It is a figure in the microprocessor which does not have a write-in path from an instruction cache to main memory showing the flow of writing processing.
[Drawing 6]It is a figure showing the example of hardware constitutions of the microprocessor which provided the command which can specify the cash for writing for every predetermined memory area.
[Drawing 7]It is a figure showing the example of hardware constitutions of the microprocessor which formed the internal flag which specifies the mode written in via an instruction cache, and the mode written in via a data cache.
[Drawing 8]It is a figure showing the example of hardware constitutions of the microprocessor which provided the command which writes in via an instruction cache.
[Drawing 9]It is a figure showing the example of the computer system with which this invention is applied.
[Description of Notations]
101 CPU
102 Instruction cache
103 Data cache
104 Main memory
105 ALU
106 Register
107 Memory unit
108 Instruction decoder

[Translation done.]